struct Data Type

/* Read Chapter 3 of the Course Notes for details. */

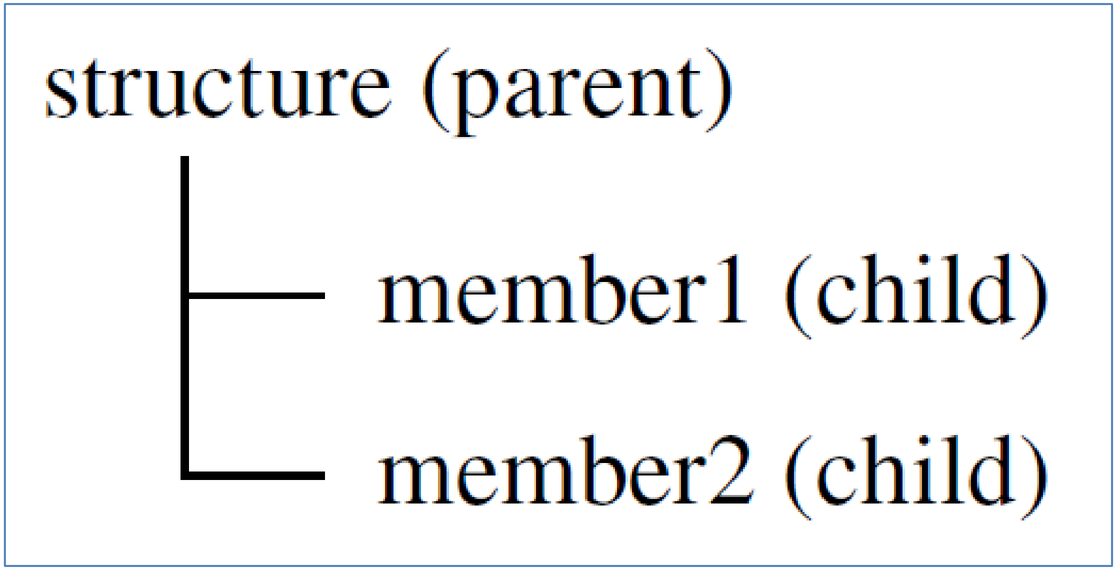
Q: What is a “structure”?

- Group of elements, called “members”, of possibly different data types (heterogeneous)
- Layman’s term for structure is “record”
- Example

  Student Record

  - ID #
  - Name (Last, First, Middle)
  - Birthday (Month, Day, Year)
  - Course
  - GPA

Graphical Representation of a Structure and Its Members (Parent-Child Relationship)

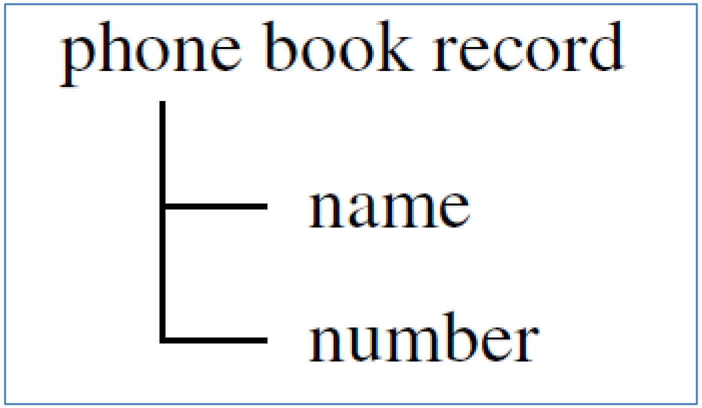


structure (parent)

member1 (child)

member2 (child)

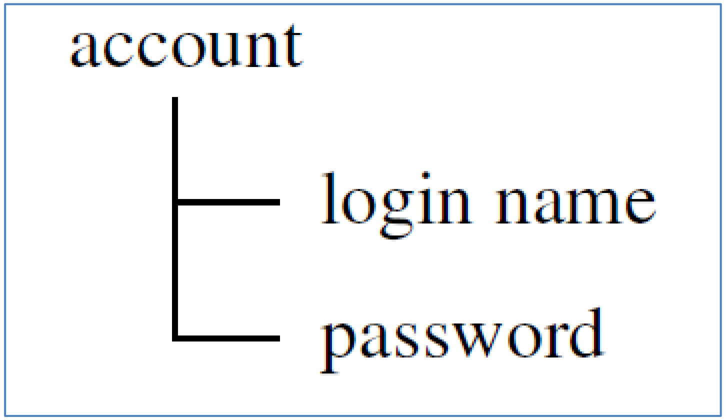
Three Graphical Examples



phone book record

name

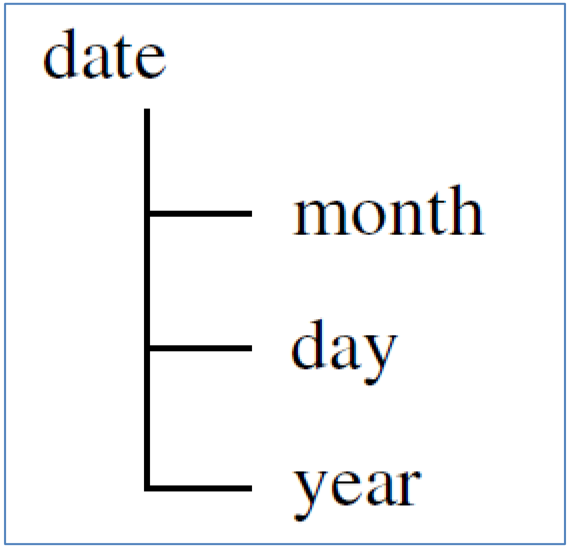
number



account

login name

password



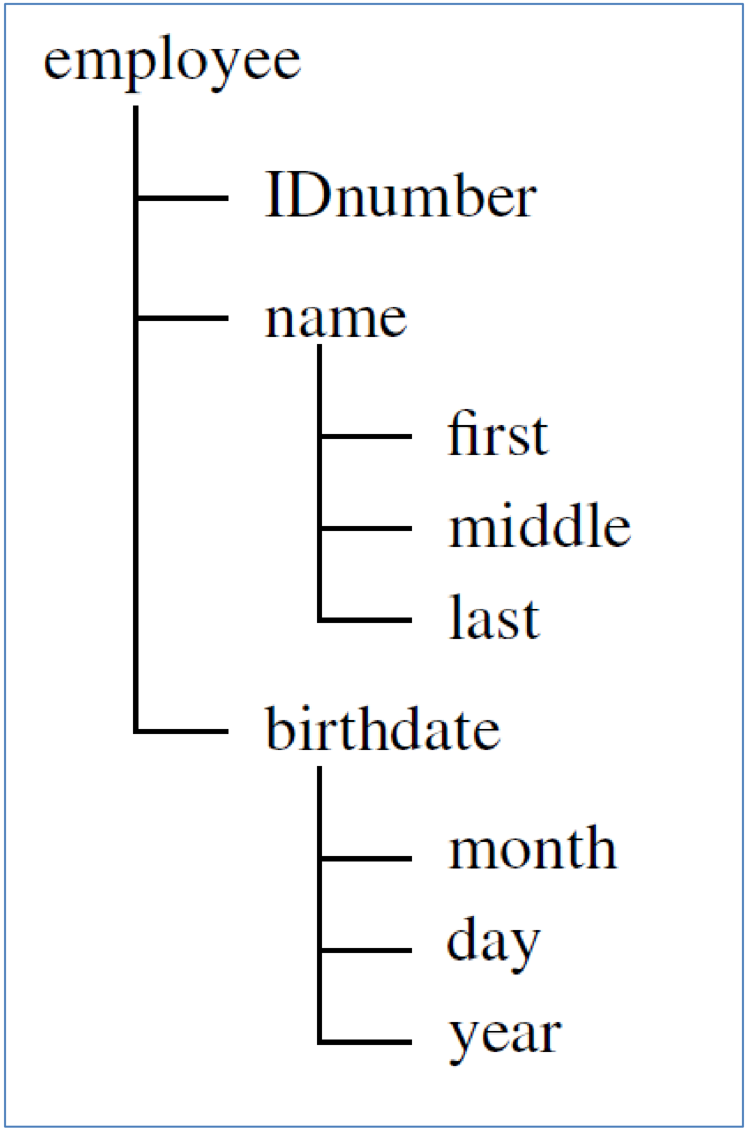
date

month

day

year

Nested Structure Graphical Example



- employee
  - IDnumber
  - name
    - first
    - middle
    - last
  - birthdate
    - month
    - day
    - year

Q: How do you declare a structure in C?

```
struct  [tag-name] {
   <data type>  <member-name>;
                 :
                 :
   <datatype> <member-name>
} [structure-var-name];
```

Variation #1 – no tag name, no structure var name

```
struct {
  char ch;
  int i;
  float f;
  double d;
};
```

- Compiler warning: “unnamed struct/union that defines no instances”
- Normally used with typedef

Variation #2 – with tag name, no struct var name

```
struct sampleTag {
  char ch;
  int i;
  float f;
  double d;
};
```

- Note: only the structure data type is declared
- Recommended practice in CCPROG2

Variation #3 – no tag name, with struct var name

```
struct {
  char ch;
  int i;
  float f;
  double d;
}   x;  // x is the instance or structure variable
```

Variation #4 –with tag name, with struct var name

```
struct  sampleTag {
  char ch;
  int i;
  float f;
  double d;
} x, y, z; // note: 3 structure variables
```

Sample Codes

```
/* 1st step: declare the structure data type */
struct sampleTag {
    char ch;
    int i;
    float f;
    double d;
};

/* 2nd step: declare the structure variable/instance */
struct sampleTag s;

/* declare two more instances on the same line*/
struct sampleTag t, u;
```

Sample Codes…

```
/* 1D array of structures */
struct sampleTag Z[100];

/* 2D array of structures */
struct sampleTag M[2][3];

/* structure pointer variable */
struct sampleTag *ptr;

/* 1D array of structure pointers */
struct sampleTag *P[10];
```

Sample Codes…

```
/* structures and functions */
struct sampleTag
A_func(struct sampleTag param)
{
   struct sampleTag temp;
   …
   return temp;
}

int main()
{
      struct sampleTag s, t;
      …
      t  = A_func(s);
      …
```

structure type function

structure type parameter

structure type local variable

Sample Codes…

```
/* structures pointers and functions */
struct  sampleTag  *
B_func(struct  sampleTag  Z[])
{
        struct  sampleTag  *pTemp;
            …
        return  pTemp;
}

int  main()
{
       struct  sampleTag  Z[10],  *ptr;
       …
       ptr   =  B_func(Z);
        …
```

structure pointer
type function

Structure Array
Name parameter

structure pointer
type local variable

Q: What operations can be performed on structures?

1. Get the memory address of the structure variable via & (address-of) operator
2. Access the structure members using the structure member operator (dot symbol)
3. Assign a structure value to a structure variable of the same data type (structure-to-structure assignment)

Q: How do you get the address of of a structure variable?

Use & operator

```
//…  assume struct sampleTag was declared above
int main()
{
    struct sampleTag s, t, u;
    struct sampleTag *ptr, *pList;

    printf(“Address of s is %p.\n”, &s);
    ptr = &t;
    pList = B_func(&u); // B_func() returns
                        // a pointer
```

Q: How do you access a structure member?

- Syntax:

Take note of the dot symbol

struct-var-name . member-name

```
// example
struct sampleTag s;
 s.ch = ‘A’;
 s.i  = 123;
 s.f = 8.5f;
 s.d = 3.1416;
```

```
// more examples
struct phoneTag {
   int number ;
   char name [30];
};

struct dateTag {
   int month ;
   int day;
   int year;
};
```

```
int main ()
{
   struct phoneTag landline;
   struct dateTag birthday;
    …
   // access members
   landline.number = 5244611;
   strcpy(landline.name, “DLSU”);
    …

   birthday.month = 10;
   birthday.day = 24;
   birthday.year = 2016;
```

// & and . operators

```
struct phoneTag {
   int number ;
    char name [30];
};

struct dateTag {
    int month ;
    int day;
    int year;
};
```

```
int main ()
{
    struct phoneTag landline;
    struct dateTag birthday;
     …
    // input data via scanf()
    scanf(“%d”, &landline.number);
    scanf(“%s”, landline.name);
     …

    scanf(“%d %d %d”,
              &birthday.month,
              &birthday.day,
              &birthday.year);
```

// . has higher priority than &

Q: How do you assign a structure to another structure?

- Syntax for structure-to-structure assignment

  struct-var-name = struct-var-name

```
// example
struct sampleTag s, t;
 s.ch = ‘A’;
 s.i  = 123;
 s.f = 8.5f;
 s.d = 3.1415;

t = s; // contents of t will be the same as s
```

```
// more examples
struct phoneTag {
   int number ;
   char name [30];
};

struct dateTag {
   int month ;
   int day;
   int year;
};
```

```
int main ()
{
   struct phoneTag landline;
   struct phoneTag office;
   struct dateTag birthday;
   struct dateTag anniversary;
   // access members
   landline.number = 5244611;
   strcpy(landline.name, “DLSU”);
    …
   office = landline;
    …
   birthday.month = 10;
   birthday.day = 24;
   birthday.year = 2016;
    …
   anniversary = birthday;
```

// INVALID example

```
struct phoneTag {
    int number ;
    char name [30];
};

struct dateTag {
   int month ;
   int day;
   int year;
};
```

```
int main ()
{
   struct phoneTag landline;
   struct phoneTag office;
   struct dateTag birthday;
   struct dateTag anniversary;
    …
   landline.number = 5244611;
   strcpy(landline.name, “DLSU”);
    …
   office = landline;
    …
   birthday.month = 10;
   birthday.day = 24;
   birthday.year = 2016;
    …
   anniversary = office; // INVALID!
```

Q: How do you specify a nested structure?

A member of a structure can be another structure. For example:

Student Record

- ID #
- Name
  - Last
  - First
  - Middle
- Birthday
  - Month
  - Day
  - Year

// declaration, 1st version

```
typedef char Str20[21];

struct studentTag {
   int ID;
   struct {
      Str20 last, first, middle;
   } name;
   struct {
     int month, day, year;
   } birthday;
};

struct studentTag s1, s2;
struct studentTag S[40]; // array of structures
```

// 2nd version

```
typedef char Str20[21];

struct nameTag {
   Str20 last, first,
middle;
 };

struct dateTag {
   int month, day, year;
};
```

```
struct studentTag {
    int ID;
    struct nameTag name;
    struct dateTag birthday;
};

struct studentTag s1, s2;
struct studentTag S[40];
```

// 3rd version with typedef

```
typedef char Str20[21];

struct nameTag {
   Str20 last, first, middle;
 };

typedef struct nametag
nameType;

struct dateTag {
   int month, day, year;
};

typedef struct dateTag
dateType
```

```
struct studentTag {
    int ID;
    nameType name;
    dateType birthday;
};

typedef struct studentTag
studentType;

studentType s1, s2;
studentType S[40];
```

```
// access members of a nested structure
struct studentTag student;

student.ID = 12345;

strcpy(student.name.last,  “TAN”);
strcpy(student.name.first, “ALEX”);
strcpy(student.name.middle, “LIM”);

student.birthday.month = 8;
student.birthday.day = 8;
student.birthday.year = 1988;
```

```
// input via scanf() members of a nested structure
struct studentTag student;

scanf(“%d”, &student.ID);

scanf(“%s”, student.name.last);
scanf(“%s”, student.name.first);
scanf(“%s”, student.name.middle);

scanf(“%d”, &student.birthday.month);
scanf(“%d”, &student.birthday.day);
scanf(“%d”, &student.birthday.year);
```

Q. Can you pass a structure as parameter?

- The value of a structure can be passed as a function parameter
- The value of the address of a structure can be passed as a function parameter
- A function can return the value of a structure as its value (i.e., the function has a struct data type)
- A function can return the address of a structure as its value (i.e., the function has a struct pointer data type)

Structure data type as parameter and return type

```
/* structures and functions */
struct sampleTag
A_func(struct sampleTag param)
{
    struct sampleTag temp;
     …
    return temp;
}

int main()
{
    struct sampleTag s, t;
     …
    t  = A_func(s);
     …
```

structure type function

structure type parameter

structure type local variable

Sample codes…

```
// assume struct sampleTag was
// already declared before the following
// functions…

void
PrintStruct_ver1(struct sampleTag s)
{
   printf(“%c %d %f %lf\n”,
               s.ch, s.i, s.f, s.d);
}

void
PrintStruct_ver2(struct sampleTag *ptr)
{
   printf(“%c %d %f %lf\n”,
               (*ptr).ch,
               (*ptr).i,
               (*ptr).f,
              (*ptr).d);
}
```

```
int
main ()
{
    struct sampleTag x;

    x.ch = ‘A’;
    x.i = 123;
    x.f = 8.8f;
    x.d = 3.1416;

    // parameter is a structure
    PrintStruct_ver1 ( x );

    // parameter is address of a struct.
    PrintStruct_ver2( &x );

    return 0;
}
```

Exercise: Implement **InputStruct()** function

```
// assume struct sampleTag was
// already declared before the following
// functions…

void
PrintStruct_ver1(struct sampleTag s)
{
     printf(“%c %d %f %lf\n”,
                  s.ch, s.i, s.f, s.d);
}

void
PrintStruct_ver2(struct sampleTag *ptr)
{
    printf(“%c %d %f %lf\n”,
                 (*ptr).ch,
                 (*ptr).i,
                 (*ptr).f,
                 (*ptr).d);
}
```

```
void
InputStruct(____________)
{
     // implement this function
}

int
main ()
{
   struct sampleTag x;

   InputStruct(__________);

   // parameter is a structure
   PrintStruct_ver1 ( x );

   // parameter is address of a struct.
   PrintStruct_ver2( &x );

   return 0;
}
```

CORRECT solution

```
// assume struct sampleTag was
// already declared before the following
// functions…

void
PrintStruct_ver1(struct sampleTag s)
{
      printf(“%c %d %f %lf\n”,
                     s.ch, s.i, s.f, s.d);
}

void
PrintStruct_ver2(struct sampleTag *ptr)
{
     printf(“%c %d %f %lf\n”,
                     (*ptr).ch,
                     (*ptr).i,
                     (*ptr).f,
                     (*ptr).d);
}
```

```
void
InputStruct( struct sampleTag *ptr)
{
      scanf(“%c %d %f %lf”,
                  &(*ptr).ch,
                  &(*ptr).i ,
                  &(*ptr).f,
                  &(*ptr).d );
}

int
main ()
{
    struct sampleTag x;

    InputStruct( &x );

                        // etc …


    return 0;
}
```

Q: How do you access a structure member indirectly?

Use the structure pointer operator denoted by

->

to access a structure member indirectly via a structure pointer variable

IMPT: can only be used with structure pointer data type

Syntax:

structure ptr var name **->** member name

Sample codes…

```
// assume struct sampleTag was
// already declared before the following
// functions…

void
PrintStruct_ver1(struct sampleTag s)
{
     printf(“%c %d %f %lf\n”,
                  s.ch, s.i, s.f, s.d);
}

void
PrintStruct_ver2(struct sampleTag *ptr)
{
     printf(“%c %d %f %lf\n”,
                 (*ptr).ch,
                 (*ptr).i,
                 (*ptr).f,
                 (*ptr).d);
}
```

```
void
PrintStruct_ver2(struct sampleTag *ptr)
{
     printf(“%c %d %f %lf\n”,
                 ptr->ch,
                 ptr->i,
                 ptr->f,
                 ptr->d);
}
```

-> makes codes on the right shorter and less prone to error

```
void
PrintStruct_ver2(struct sampleTag *ptr)
{
    printf("%c %d %f %lf\n",
               (*ptr).ch,
               (*ptr).i,
               (*ptr).f,
               (*ptr).d);
}

void
InputStruct( struct sampleTag *ptr)
{
     scanf("%c %d %f %lf",
              &(*ptr).ch,
              &(*ptr).i ,
              &(*ptr).f,
              &(*ptr).d );
}
```

```
void
PrintStruct_ver2(struct sampleTag *ptr)
{
    printf("%c %d %f %lf\n",
               ptr->ch,
               ptr->i,
               ptr->f,
               ptr->d);
}

void
InputStruct( struct sampleTag *ptr)
{
     scanf("%c %d %f %lf",
              &ptr->ch,
              &ptr->i ,
              &ptr->f,
              &ptr->d );
}
```

Accessing elements of a list of structures

```
// allocate space (static mem. alloc.)
struct sampleTag A[10];

// print the members of A[i]
printf("%c %d %f %lf\n",
        A[i].ch,
        A[i].i,
        A[i].f,
        A[i].d);
```

```
//  recall that:   A[i] == *(A + i)
// print the members of A[i]
printf("%c %d %f %lf\n",
        ( *(A + i) ).ch,
        ( *(A + i) ).i,
        ( *(A + i) ).f,
        ( *(A + i) ).d);

// use -> instead of * and .
printf("%c %d %f %lf\n",
        (A + i)->ch,
        (A + i)->i,
        (A + i)->f,
        (A + i)->.d);
```

A[i].ch is equal to **(*(A + i).ch)** is equal to **(A + i)->ch**

-- The End --